sification and a step-by-step algorithm for the implementation of this machine learning task is given. The technical implementation of the system for detecting financial fraud with payment cards based on Microsoft Azure cloud services is developed and substantiated. The effectiveness of the proposed system for detecting fraudulent transactions is assessed, where sensitivity and specificity are selected as the criteria for efficiency being generally accepted indicators in machine learning theory.
Keywords: financial fraud detection systems, machine learning, cloud services, decision support systems, security of operations.

Фесенко Андрій Олексійович, кандидат технічних наук, асистент кафедри кібербезпеки та захисту інформації факультету інформаційних технологій КНУ імені Тараса Шевченка.
E-mail: aafesenko88@gmail.com.
Orcid ID: 0000-0001-5154-5324.
Фесенко Андрей Алексеевич, кандидат технических наук, асистент кафедры кибербезопасности и защиты информации факультета информационных технологий КНУ имени Тараса Шевченко.
Fesenko Andrii, PhD, assistant of the Cybersecurity and Information Security Department of the Information Technology Faculty, Taras Shevchenko National University of Kyiv.

Папірна Ганна Костянтинівна, студентка кафедри кібербезпеки та захисту інформації факультету інформаційних технологій КНУ імені Тараса Шевченка
E-mail: mhiaofy@gmail.com.
Orcid ID: 0000-0001-9989-9412.
Папирна Анна Константиновна, студентка кафедры кафедры кибербезопасности и защиты информации факультета нформационных технологий КНУ имени Тараса Шевченко.
Papirna Hanna, student of the Cybersecurity and Information Security Department of the Information Technology Faculty of Taras Shevchenko National University of Kyiv.

Бауиржан Мадіна Бауиржанівна, докторант PhD, Казахський національний дослідницький технічний університет ім. К.І. Сатпаєва, Алмати, Республіка Казахстан.
E-mail: madina890218@gmail.com.
Orcid ID: 000-0002-8287-4283.
Бауыржан Мадина Бауыржановна, докторант PhD, Казахский национальный исследовательский технический университет им. К.И. Сатпаева, Алматы, Республика Казахстан.
Bauyrzhan Madina, PhD Student, Satbayev University, Almaty, Republic of Kazakhstan.

DOI: 10.18372/2410-7840.21.13768
УДК 004.274:004.056

ПОБУДОВА СКІНЧЕННИХ АВТОМАТІВ РЕКОНФІГУРОВНИМИ ЗАСОБАМИ ДЛЯ ВИРІШЕННЯ ЗАДАЧ ІНФОРМАЦІЙНОЇ БЕЗПЕКИ

Сергій Гільгурт

Протидія загрозам інформаційної безпеки потребує вживання заходів різного плану – організаційних, технічних, криптографічних тощо. Мережеві системи виявлення вторгнень, антивіруси, та інші засоби технічного захисту, робота яких заснована на використанні сигнатур, мають вирішувати в реальному часі обчислювально складну задачу множинного розпізнавання рядків, яка на відміну від одиночного розпізнавання має метою одночасний пошук у вхідних даних великої кількості зразків. Програмні рішення вже не впорюються з цією проблемою через сталий зріст об'єму мережевого трафіку, кількості та складності атак. Тому все більшого поширення набувають апаратні рішення з використанням реконфігурованих пристроїв на базі ПЛІС типу FPGA, які поєднують в собі близьку до апаратної продуктивність із гнучкістю програмного забезпечення. На сьогодні найбільш поширеними є три підходи щодо побудови апаратних схем множинного розпізнавання, робота яких заснована на використанні: асоціативної пам'яті, фільтра Блума та скінченних автоматів. Кожен з підходів має свої власні переваги та вади. Ефективна побудова все більш складних сигнатурних засобів технічного захисту інформації, яки б відповідали вимогам оптимального функціонування в залежності від зовнішніх умов, неможлива без всебічному аналізу властивостей та специфічних рис кожного з підходів. У цьому дослідженні з метою підвищення ефективності створюваних на базі ПЛІС засобів захисту інформації проаналізовані переваги та недоліки третього з перелічених підходів. На основі аналізу світового досвіду використання скінченних автоматів також досліджені особливості їх реалізації на ПЛІС, проблеми, що виникають, та шляхи їх вирішення.
Ключові слова: *захист інформації, сигнатурний аналіз, ПЛІС, скінченний автомат, алгоритм Ахо-Корасік, ефективність.*

Вступ

У зв'язку з припиненням зростання частоти традиційних процесорів, а також через сталий зріст кількості та складності зловмисної активності в інформаційних системах програмна реалізація мережевих систем виявлення вторгнень (МСВВ), антивірусів, засобів протидії мережевим хробакам та інших додатків, робота яких заснована на сигнатурному аналізі, вже не відповідає вимогам щодо швидкодії. Тому розробники вимушені використовувати апаратні платформи на основі програмованих логічних інтегральних схем (ПЛІС) типу FPGA. Висока продуктивність реконфігуровних засобів у поєднанні з гнучкістю, близької до програмної, якнайкраще відображає динамічну природу галузі інформаційного захисту. Сприяє застосуванню ПЛІС також наявність и розвиненість ринку реконфігуровних обчислювачів на їх основі, які можуть бути успішно використані в якості апаратної платформи [1].

Але зростання складності завдань комп'ютерної безпеки триває, як і збільшення об'ємів мережевого трафіку. Тому підвищення ефективності реконфігуровних апаратних засобів захисту інформації є *актуальною науково-технічною проблемою*.

В якості можливого шляху вирішення цієї проблеми у роботі [2] запропонований метод створення універсальної комбінованої структури модуля розпізнавання МСВВ, суть якого полягає у поєднанні в одному пристрої переваг різних підходів до побудови схем розпізнавання на ПЛІС. Для його успішної реалізації ключовим чинником є детальне розуміння технічних властивостей кожного з напрямів, що комбінуються. Тому актуальним завданням є також *глибокий аналіз особливостей відомих підходів до побудови апаратних схем розпізнавання на базі ПЛІС*.

При створенні реконфігуровних засобів інформаційного захисту найкращі здібності продемонстрували підходи, що засновані на використанні:

– асоціативної пам'яті та цифрових компараторів;

– фільтрів Блума, що використовують хеш-функції;

– цифрових автоматів, а саме скінченних автоматів.

В попередніх роботах автора [3] та [4] розглянуті перший та другій підхід відповідно.

Дане дослідження присвячено всебічному аналізу третього з перелічених напрямів. При цьому використовується багатий досвід, здобутий чисельними розробниками реконфігуровних засобів інформаційного захисту зі всього світу. В процесі вивчання предмету дослідження, як і в попередніх роботах, головна увага була спрямована на:

– особливості (переваги та недоліки) підходу, що вивчається, в сенсі забезпечення показників ефективності, сформульованих нижче;

– специфіку реалізації підходу на ПЛІС;

– складнощі та проблеми, що виникають під час створення реконфігуровних засобів і шляхи їх подолання з метою покращення показників ефективності.

Для обґрунтованого порівняння різних підходів та технічних рішень щодо побудови реконфігуровних засобів захисту інформації, потрібно визначитися з показниками. Історично склалося, що мережеві системи виявлення вторгнень стали першими сигнатурними засобами інформаційного захисту, при побудові яких почалися використовуватися ПЛІС. Тому, не втрачаючи загальності міркування, сформулюємо показники ефективності, що висуваються при побудові з використанням реконфігуровних засобів саме МСВВ.

1. Показники ефективності реконфігуровних МСВВ. Показники ефективності сигнатурних реконфігуровних засобів інформаційної безпеки можна умовно поділити на:

– вартісні;

– швидкісні (або показники продуктивності);

– функціональні.

До вартісних показників належать обсяги логічних ресурсів програмованої логіки, які задіяні для створення цифрової схеми, витрати на пам'ять (як зовнішню відносно кристалу ПЛІС, так і внутрішню – блочну пам'ять BRAM та розподілену у вигляді тригерів логічних комірок), а також загальна вартість володіння, яка включає витрати на розробку, виготовлення, програмування та експлуатацію системи.

До параметрів продуктивності відносять об'єм словнику сигнатур, які розпізнає засіб, пропускну здатність, а також передбачуваність пропускної здатності.

До функціональних показників відносяться спроможність системи МСВВ працювати у режимі запобігання вторгнень, здатність до динамічного оновлення словнику сигнатур без припинення процесу розпізнавання, здатність протидіяти атакам на МСВВ тощо.

Важливим проміжним показником, який пов'язує швидкісні характеристики з вартісними, є масштабованість – здатність нарощувати продуктивні здібності без завеликих додаткових ресурсних

витрат. Розрізняють масштабованість за пропускною здатністю, за об'ємом словнику сигнатур та за довжиною патернів – шуканих послідовностей символів.

Саме на ці показники ефективності будемо орієнтуватися при аналізі підходу до побудови реконфігуровних засобів захисту, який базується на використанні скінченних автоматів.

2. Скінченні автомати. Для опису поведінки та створення складних дискретних систем вже багато років успішно використовують математичний апарат цифрових (або дискретних) автоматів [5].

Абстрактний автомат (АА), відповідний англомовний термін – state machine (SM), у загальному випадку задається шісткою елементів:

$$A = \{X, Y, S, S_0, f_s, f_y\}, \quad (1)$$

де X – множина вхідних сигналів; Y – множина вихідних сигналів; S – множина станів автомата; S_0 – початковий стан автомата; f_s – функція переходів з одного стану в інший; f_y – функція виходів автомата.

Класичний АА або цифровий автомат (ЦА) в початковий момент часу перебуває у стані S_0; на кожному такті в залежності від вхідних сигналів переходить з одного стану в інший згідно функції переходів f_s; при цьому його вихідні сигнали формуються згідно функції виходів f_y. Якщо функція виходів залежить тільки від поточного стану, ЦА називають автоматом Мура, якщо від поточного стану та вхідних сигналів – автоматом Мілі. Звернемо увагу, що в обох випадках вихід змінюється при кожному переході. Автомат такого класу, називають *перетворювачем* або *трансдуктором* (transducer) [6]. Подібні ЦА були давно відомі та активно використалися при створенні різноманітних цифрових пристроїв та систем в роки становлення та розвитку цифрової обчислювальної техніки.

Але останнім часом теоретичні дослідження та практичні розробки частіше використовують інші класи ЦА. Так автомат – *акцептор* (acceptor, recognizer), також відомий як *виявлювач послідовностей* (sequence detector) видає на виході активний сигнал тільки в разі, якщо на його вхід надійшла певна послідовність певних комбінацій вхідних сигналів – символів. На практиці функція акцептора полягає в виявленні (розпізнаванні) певного слова (послідовності символів) у вхідному потоці даних. В термінах теорії обчислень на строках [7] акцептор виявляє входження підрядка в рядок. Технічно акцептори реалізуються таким чином, що в результаті надходження на вхід автомата питомої послідовності символів він переходить у так званий *прийнятний* стан. Інакше кажучи, сигнал на виході автомата відсутній поки здійснюються переходи між неприйнятними станами та з'являється тільки в разі досягнення прийнятного стану.

Коли потрібно виявляти у вхідному потоці даних замість одиничного підрядку одночасно певну множину зразкових слів (або *патернів*), тобто необхідно виконувати задачу множинного розпізнавання рядків, множина станів автомата містить декілька прийнятних станів. В разі досягнення будь-якого з них автомат сигналізує про розпізнавання відповідного патерну. Такий автомат називають *розпізнавачем* (classifier). Саме такі автомати використовуються при створенні сигнатурних засобів технічного захисту інформації, у тому числі – МСВВ.

Фактично, автомат – розпізнавач є різновидом класичного цифрового автомату (1). Відмінність полягає у відсутності множини вихідних сигналів Y та функції виходів f_y, замість яких з'являється множина прийнятних станів F:

$$A = \{X, S, S_0, f_s, F\}. \quad (2)$$

Отже автомат для множинного розпізнавання рядків описується не шісткою, а п'ятіркою відповідних елементів.

Якщо у виразах (1) та (2) елементи X, Y, S та F є скінченними множинами, відповідні автомати називають *скінченними автоматами* (СА). Це стосується будь-якого дискретного автомата. Але за спостереженням автора в публікаціях останніх років, особливо в англомовних, у переважній більшості випадків словосполука *скінченний автомат* (finite state machine (FSM) або finite automaton (FA)) замість виразу *цифровий автомат* (state machine (SM)) використовується саме для посилання на автомати, що розпізнають рядки, тобто на автомати – розпізнавачі. Тому в даному дослідженні для позначення автоматів, на яких будуються сигнатурні засоби технічного захисту інформації, також вживатиметься цей термін – "скінченними автомат".

Оскільки в автоматах – розпізнавачах на відміну від класичних ЦА виду (1) замість вхідних сигналів вживаються символи, тобто групи сигналів (кількістю у вісім одиниць, якщо символ кодується одним байтом, або іншого розміру, кратного восьми, якщо кодування багатобайтове), то множина X набуває сенсу *вхідного алфавіту* автомата (множини символів, яки можуть бути подані на вхід).

За функціонуванням СА дещо відрізняється від цифрового автомата (1). В початковий момент часу СА перебуває у стані S_0; на кожному такті отримує вхідний символ (з алфавіту X) і в залежності від його значення переходить з одного стану в інший згідно функції переходів f_s; досягнувши

одного з прийнятних станів, відає сигнал про розпізнавання відповідного слова (підрядка), після чого продовжує роботу з поточного стану.

Скінченні автомати бувають *детермінованими* (ДСА) та *недетермінованими* (НСА). Відповідні англомовні терміни – *deterministic finite automaton* (DFA) та *nondeterministic finite automaton* (NFA). У детермінованому автоматі на кожному такті можливий лише один перехід лише до одного стану. Як наслідок в кожний окремий момент часу ДСА може перебувати тільки в одному стані. Для НСА вказані обмеження не виконуються. Теоретично довільний НСА може бути трансформований в еквівалентний за функціональністю але значно більш складний за кількістю станів ДСА. Але на практиці ДСА та НСА в наслідок особливостей технічної реалізації мають різні сфери застосування.

3. Алгоритм Ахо-Корасік та його реалізація у вигляді скінченного автомата. Алгоритм Ахо-Корасік (АК) [8] є зразком класичного засобу множинного розпізнавання рядків на відміну від одношаблонних алгоритмів, орієнтованих на виявлення одиночних патернів. Його суть полягає у тому, що з наданого набору патернів за певними правилами створюються детермінований скінченний автомат – розпізнавач. Цей алгоритм набув значного поширення серед розробників сигнатурних систем інформаційного захисту. Не зважаючи на те, що він був одним з перших алгоритмів множинного розпізнавання, він досі залишається одним з найзатребуваніших інструментів сигнатурного аналізу. В англомовній літературі навіть сформувався усталений термін – абревіатура AC-DFA (Aho-Corasick deterministic finite automaton).

Існує безліч джерел, в яких цей алгоритм та особливості його використання докладно описані, в тому числі при реалізації на ПЛІС [9 – 14, 16 – 20]. Але, як буде пояснено нижче, певні моменти є суттєвими для порівняння показників ефективності різних підходів до побудови схем апаратного розпізнавання. Тому розглянемо алгоритм Ахо-Корасік, звертаючи увагу на такі особливості.

Рис. 1 ілюструє процес створення ДСА згідно алгоритму АК для розпізнавання трьох патернів: "SHIP", "HIS" та "IN" [9].

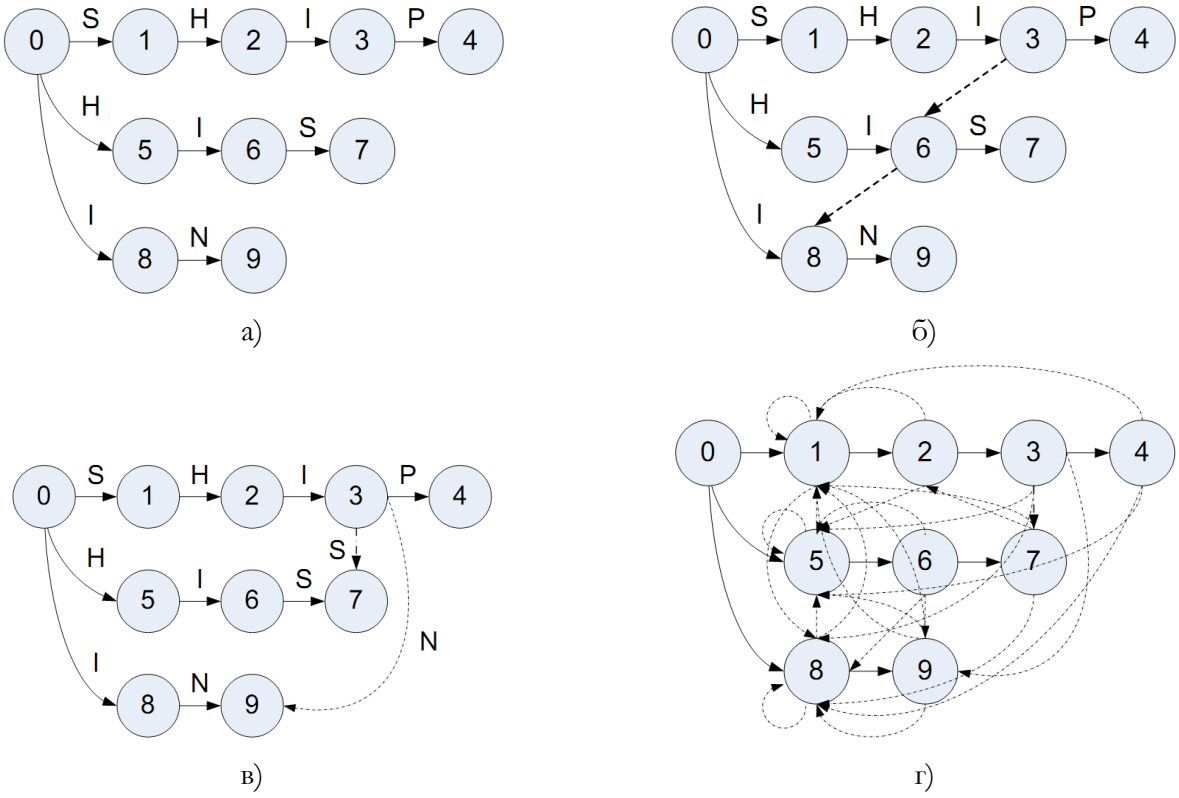

Рис. 1. Формування ДСА згідно алгоритму АК для множини патернів {SHIP, HIS, IN}

Процедура створення автомата згідно алгоритму АК складається з вдвох кроків.

На першому кроці (рис. 1, а) будується ациклічний орієнтовний граф у вигляді дерева, коренем якого є початковий стан S_0. Кожен з патернів, що підлягають розпізнаванню, додає до графу ланцюжок вершин, що відповідають станам автомату, таким чином, що кожний символ відзначає дугу до наступної вершини – стану, починаючи з кореня аж до кінця патерну. Фактично, вже той ав-

томат, структуру якого зображено на рис. 1, а, здатен розпізнавати відповідні патерни, якщо його функція переходів f_s визначена наступним чином: в разі надходження на вхід автомата символу, який відповідає дузі, він переходить до стану, на який ця дуга вказує, а у випадку надходження будь-якого іншого символу, здійснює перехід до початкового стану (ці зворотні переходи не відображені на рисунку, щоб не ускладнювати його). Стани S_4, S_7 та S_9 цього скінченого автомату є прийнятними станами; досягнення будь-якого з них свідчить про наявність у вхідній послідовності підрядка "SHIP", "HIS" або "IN" відповідно.

Але такий автомат за своєю продуктивністю не відрізняється від сукупності окремих автоматів, кожен з яких здійснює одиночне розпізнавання свого патерну. З іншого боку, базам даних сигнатур СВВ та антивірусів притаманна властивість самоподоби, суть якої полягає у тому факті, що багато патернів мають спільні фрагменти. У нашому прикладі підрядки "SHIP" та "HIS" мають спільний фрагмент "HI". Надмірність словнику сигнатур, що обумовлена даною властивістю, дозволяє скоротити кількість операцій порівняння, відтак прискорити процес розпізнавання. Саме здібність до використання цієї надмірності задля підвищення швидкодії відрізняє алгоритми множинного розпізнавання від одношаблонних алгоритмів.

Щоб використати подібність патернів "SHIP" та "HIS", потрібно змінити функцію переходів таким чином, щоб перебуваючи у стані S_3 при надходженні символу, відмінного від "P", автомат не робив перехід у початковий стан, а опинявся у стані S_6. У загальному випадку нове правило переходів формулюється наступним чином. Якщо розпізнавання поточного патерну переривається, треба перевірити, чи не є вже розпізнаний фрагмент префіксом (декількома початковими символами) іншого патерну. Якщо таких патернів виявиться більш одного, обирається той, з префіксом якого збігається найбільша кількість вже розпізнаних патернів. В розглянутому прикладі в стані S_3 автомат частково розпізнав не тільки послідовність "HI", яка збігається с префіксом патерну "HIS", але й односимвольну послідовність "I", яка збігається с префіксом патерну "IN". Але фрагмент "HI" довше фрагменту "I", тому перехід слід робити у стан S_6. Але стрибок зі стану S_3 у стан S_6 повинен здійснюватися одночасно з переходом зі стану S_6 у наступний стан в залежності від вхідного символу, який поступає за розпізнаним тактом раніше символом "I", тобто автомат повинен здійснювати два переходи на один прийом вхідного символу. У детермінованому автоматі це неприпустимо. Тому алгоритм АК об'єднує перехід на ланцюжок подібного патерну з тактом розпізнавання наступного символу. В результаті перехід зі стану S_3 замість стану S_6 здійснюється у стан S_7, але за умови подання на вхід автомату символа "S" (див. рис. 1, в). За таким принципом можна використати також частково розпізнаний фрагмент "I" в разі надходження символу "N", тобто додати в функцію переходів дугу зі тану S_3 в стан S_9. Пошук таких переходів виконується на другому кроці побудови автомату АК.

Насправді властивість самоподоби для обраних трьох патернів не обмежується спільними фрагментами "HI" та "I". Наприклад, якщо у вхідній послідовності за префіксом якогось патерну слідує знов такій же префікс, здійснюється перехід в початок ланцюжку даного патерну, тобто реалізується ефект самоподоби всередині самого патерну. На рис. 1, г наведені всі можливі переходи, які повністю реалізують властивість самоподоби набору з трьох патернів "SHIP", "HIS" та "IN". При цьому задля більшої наочності поряд з дугами опущені відповідні літери, також опущені хибні переходи у початковий стан. Як можна бачити, навіть для такого простого прикладу загальна кількість переходів у повністю побудованому автоматі АК доволі велика.

Наразі проаналізуємо різновиди переходів в алгоритмі АК та проведемо їх класифікацію.

Можна розрізнити чотири типи переходів у сформованому автомати АК:

1. *Прямі* (direct, basic або goto) *переходи* – ті, які закладаються в граф на першому кроці (рис. 1, а).

2. *Перехресні* (cross) *переходи*, яки використовують надмірність внаслідок ефекту самоподоби (рис. 1, в).

3. *Хибні* (failure) *переходи* – переходи у початковий стан.

4. *Післястартові* (restartable [10]) *переходи* – ті, що здійснюються на стани, що безпосередньо слідують після початкового стану (на рис. 1, в це переходи з будь-якого стану в стани S_1, S_5 та S_8).

Наявність четвертого типу переходів обумовлена тим фактом, що остання літера майже кожного з патернів збігається з першою літерою (односимвольним префіксом) якогось іншого патерна. Відокремлення переходів даного типу від інших перехресних переходів обумовлено особливостями технічної реалізації автоматів АК, як буде пояснено пізніше.

4. Базова схема. Загальна структура апаратної реалізації алгоритму Ахо-Корасік подана на рис. 2.

Його основу складає запам'ятовуючий пристрій (ЗП) для зберігання таблиці переходів скінченого автомата [11, 12]. Кожна комірка ЗП містить номер наступного стану та вектор збігу. До значення номеру наступного стану, що витягується з пам'яті, шляхом конкатенації додається код символу із вхідній послідовності. Отримане значення подається на адресний вхід запам'ятовуючого пристрою та обирає відповідний рядок інформації. Для рядків з прийнятними станами вектор збігу містить одиницю у позиції, що позначає відповідний патерн.

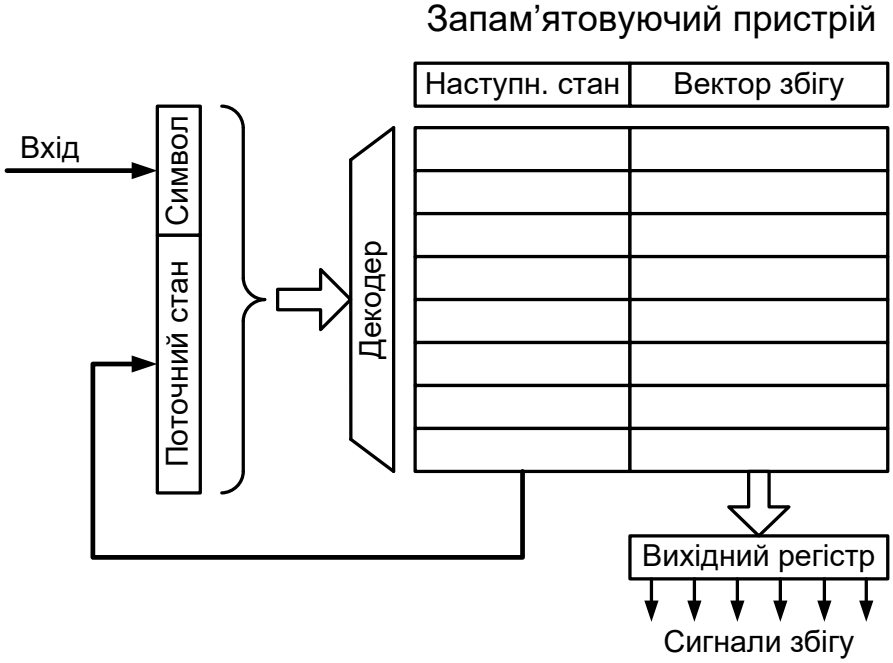

Рис. 2. Структурна схема типової реалізації скінченого автомата АК

5. Головні властивості базової схеми.

Найважливіша перевага схеми – *передбачуваність пропускної здатності* та її незалежність від об'єму словнику сигнатур і від особливостей патернів, зокрема від їх довжин. Теоретично, скінченний автомат приймає один символ з вхідної послідовності за кожен такт. Але на практиці якщо розмір таблиці переходів завеликий, і для її зберігання потрібно використовувати зовнішню відносно ПЛІС пам'ять, можливі ситуації, коли кожне звернення до ЗП потребуватиме декілька тактів. Це окрім того, що зовнішня пам'ять сама по собі повільніша за внутрішню.

Ще одна важлива перевага підходу до множинного розпізнавання рядків з використанням скінченного автомату – це *здатність до динамічного оновлення словнику сигнатур* без припинення процесу розпізнавання. Для зміни алгоритму роботи СА достатньо обновити вміст ЗП. Якщо кількість станів нового алгоритму не перевищує розмір запам'ятовуючого пристрою, це можна зробити, фактично, без затримки процесу функціонування СА. Як було зазначено в попередніх оглядах [3, 4], фільтр Блума для реалізації цієї функції потребує ускладнення його структури шляхом заміни комірок пам'яті на лічильники, а асоціативна пам'ять на цифрових компараторах взагалі не дає змоги здійснювати динамічне оновлення.

Стосовно споживання ресурсів, як можна бачити, СА потребує незначну кількість ресурсів логіки, обмежуючись, фактично, лише схемою керування та контролером ЗП. З іншого боку, об'єм запам'ятовуючого пристрою може сягати значних величин. Оскільки в загальному випадку кількість станів дорівнює загальній кількості символів у словнику, а кількість переходів з кожного стану в інший в граничному випадку може сягати 256 (при побайтовій обробці), то безпосередня реалізація схеми згідно рис. 2 потребує кількість байтів пам'яті, що чисельно дорівнює:

$$M = n \cdot m \cdot 256 \cdot (1 + n / 8), \qquad (1)$$

де n - кількість патернів у базі сигнатур, а m – їх середня довжина. При розмірах словнику в десятки та сотні тисяч патернів та їх середній довжині в десятки символів розмір ЗП сягатиме порядків сотень гігабайт – десятків терабайтів.

Отже, підхід характеризується високою ресурсоємністтю, а саме – *значними витратами на пам'ять*, що вважається його головним недоліком.

Тому переважна більшість досліджень щодо застосування СА АК та відповідних технічних рішень спрямована саме на зменшення об'єму вживаної пам'яті.

Слід зауважити, що квадратична залежність (1), яка призводить до лавиноподібного зросту необхідних ресурсів при збільшенні кількості патернів, означає *погану масштабованість за об'ємом словнику сигнатур*.

Стабільна, проте відносно *низька пропускна здатність* СА АК також вважається її недоліком.

Але більш суттєвим недоліком є складність підвищення швидкодії базової схеми автомата. Тобто підхід також має *погану масштабованість за пропускною здатністю*.

З іншого боку, алгоритм Ахо-Корасік має *добру масштабованість за довжиною патернів*.

6. Особливості реалізації схеми АК на ПЛІС. Як було вказано вище, значну частину скінченного автомата становить запам'ятовуючий пристрій, тому основні складнощі, що виникають при побудові на програмованій логіці СА АК для розпізнавання рядків, полягають в створенні цього пристрою та організації ефективного обміну даними з ним. В разі використання реконфігуровних обчислювачів проблема загострюється в зв'язку з тим, що бортова пам'ять цих пристроїв має фіксовану структуру, не завжди зручну для побудови ЗП для скінченного автомата. В роботі [13] детально розглянуті питання організації роботи з пам'яттю при побудові засобів розпізнавання на основі алгоритму Ахо-Корасік для задач захисту інформації. Ретельно вивчена функціональність, яка потрібна для ефективного доступу до даних, враховуючі можливість повторного використання (кешування) інформації. Досліджені особливості взаємодії з пристроями динамічної пам'яті різних поколінь. Доведена можливість практичного досягнення високих значень пропускної здатності.

7. Зменшення ресурсних витрат.

Як слідство головного недоліку підходу на основі скінченних автоматів, більшість досліджень щодо апаратного застосування алгоритму Ахо-Корасік, присвячена проблемі зменшення вживання ресурсів пам'яті.

Перш за все багато розробників пропонують різноманітні техніки зменшення об'єму ЗП за рахунок кодування таблиці переходів. Цій шлях нібито дозволяє досягнути мети, але за рахунок виконання додаткових операцій, тобто не відрізняється ефективністю. Більш результативними виявляються інші напрями досліджень.

В роботах [11, 12] запропонована модифікація алгоритму АК таким чином, щоб використовувати збіг не тільки суфіксів одних патернів с префіксами інших, а також – подібних інфіксів (середніх частин) різних патернів. Експериментальна реалізація підтвердила життєздатність даної ідеї, показав можливість скорочення пам'яті на 21%.

Використання наведеної вище (в підрозділі 3) класифікації переходів в алгоритмі АК дозволяє більш витончено оперувати показниками ефективності.

Відомо, що значну частку переходів, а отже, і пам'яті, займають хибні переходи у початковий стан, а також післястартові переходи. Спеціальна техніка на основі використання пріоритетів дозволила автору дослідження [14] скоротити кількість переходів цих двох класів до 256 одиниць для довільного за розміром словника патернів. (Ось чому було доцільно виокремити післястартові переходи з загального числа перехресних переходів). Остання чисельна група переходів, що залишаються не оптимізованими – це основні перехресні переходи. Згідно дослідженню [10] в автоматі АК, побудованому по базі даних сигнатур відомої відкритої системи MCBB Snort, частка таких переходів на момент проведення дослідження становила 79,2%; для антивірусній системи ClamAV ця доля була ще більша – 95,9%. Автори цієї ж роботи модифікували схему СА АК, створивши так званий кешований детермінований скінченний автомат Cached deterministic finite automaton (CDFA). Вони стверджують, що розробка дозволяє видалити до 90% переходів в апаратних додатках інформаційного захисту мережевих об'єктів. В результаті авторам роботи вдалося скоротити ресурси пам'яті до 81 Kb для бази сигнатур Snort об'ємом в 1800 записів, та до 29 Kb для бази ClamAV розміром в 50000 записів.

8. Обробка більш ніж по одному символу за такт.

Окрім надмірного споживання ресурсів пам'яті СК АК характеризується не дуже гарними швидкісними показниками. Як будь-який скінченний автомат він обробляє вхідну інформацію принципово послідовно – символ за символом. У роботі [15] зроблена спроба прискорити роботу алгоритму Ахо-Корасік шляхом обробки більш, ніж одного байту за такт. При цьому СА розглядає декілька байтів разом як один символ алфавіту X. В іншому принцип роботі СА не змінюється. Оскільки заздалегідь невідомо, з яким зсувом опиниться питомий патерн у вхідному потоці даних,

потрібно організувати паралельну роботу відповідної кількості автоматів, які розглядають скомбіновані патерни с різними зміщеннями.

На рис. 3 наведено схему, яка обробляє по три символи за такт.

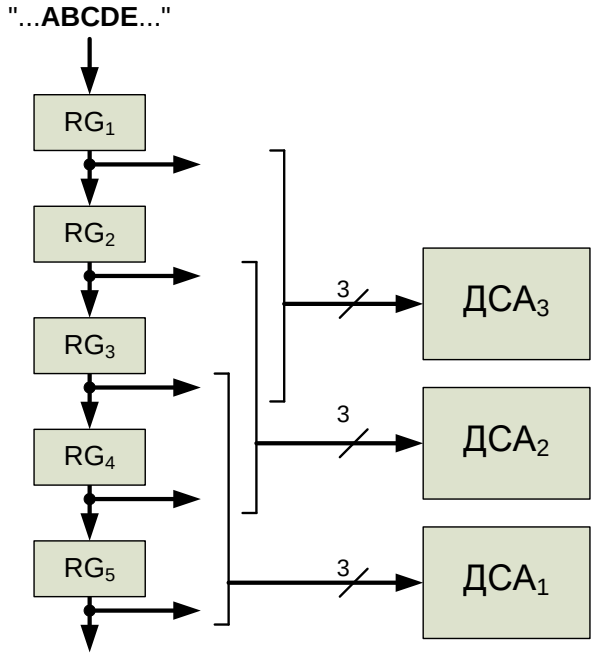

Рис. 3. Схема багатосимвольного розпізнавання

Вхідна інформація пересувається у конвеєрі, що складається з регістрів RGi. Кожен з ДСА отримує на вході по три символи за такт. Очевидно, що така схема не дуже ефективна. По-перше, кожен з автоматів потребує значно більше ресурсів пам'яті, оскільки його алфавіт складається не з 256, а з 2^{24} символів. По-друге, таких автоматів потрібно три. Тобто масштабованість за пропускною здатністю виявляється дуже поганою. Авторам роботи [15] вдалося за рахунок запропонованих методів суттєво покращити ситуацію та досягнути трикратного прискорення для модифікованої схеми, але загальний результат виявився не дуже оптимістичним.

9. Небайтова розрядність обробки даних.

Оригінальне рішення було запропоновано в роботі [16], що було назване авторами як алгоритм "bit-split". Його суть полягає в заміні одного автомата, що обробляє 8-бітні символи на декілька паралельно працюючих підавтоматів (tiny automata), кожен з яких аналізує по 1, 2 або 4 біти. За рахунок зменшення "символів", що обробляються, суттєво скоротився розмір алфавіту. Наприклад для 2-бітної обробки кількість допустимих символів в алфавіті скоротилася до 16 (замість 256). Відповідно скоротилася кількість можливих переходів з кожного стану. Ця техніка виявилася вдалою, в неї з'явилися послідовники, яки допрацювали та розвили її. В роботі [17] було доведено, що оптимальним розбиттям є ділення саме по 2 біти. В дослідженні [18] за допомогою попередньої обробки фільтром Блума вдалося позбавитися зайвих переходів в підавтоматах. В роботі [19] досліджена оптимальна кластеризація бази даних сигнатур між bit-split підавтоматами.

Автори розробки [12], яка згадувалася раніше, дослідили сумісність своєї модифікації алгоритму АК щодо використання збігу інфіксів з технікою bit-split. Виявилося, що їх розробка незалежна відносно даної техніки та дозволяє додати ще 24% економії ресурсів пам'яті до оригінальної схеми [16].

10. Конвеєризація

В роботі [20] була вперше запропонована, а потім розвинута в [9] техніка застосування конвеєризації при побудові СА АК. Традиційно в обчислювальній техніці конвеєр використовується для підвищення пропускної здатності. Але у випадку алгоритму Ахо-Корасік цей метод обробки інформації дозволив також суттєво скоротити потрібні ресурси пам'яті. Автори [9] формально довели, що лінійний конвеєр із H ступенів дозволяє видалити всі перехресні переходи в графі АК від початкового стану до рівня H.

Висновки

Підсумуємо отримані результати щодо особливостей та показників ефективності розглянуто підходу.

Головні переваги підходу на основі алгоритму Ахо-Корасік – це *передбачуваність пропускної здатності*, *добра масштабованість за довжиною патернів* та *незначне споживання логічних ресурсів* ПЛІС, а також *здатність до динамічного оновлення* словнику сигнатур без припинення процесу розпізнавання.

Головні вади – *надмірне споживання ресурсів пам'яті*, *погана масштабованість за об'ємом словнику сигнатур* та *за пропускною здатністю*. Але чисельні дослідження дозволили суттєво пом'якшити цю проблему.

Недоліком також є відносно *невисока пропускна здатність* порівняно з асоціативною пам'яттю та фільтром Блума.

Здобуті в даному дослідженні відомості дозволять розробникам створювати більш ефективні реконфігуровні засоби інформаційної безпеки.

ЛІТЕРАТУРА

[1]. С. Гильгурт, "Реконфигурируемые вычислители. Аналитический обзор", *Электронное моделирование,* 2013.

[2]. С. Гильгурт, "Применение реконфигурируемых вычислителей для аппаратного ускорения сигнатурных систем защиты информации", *Моделювання-2018*, Київ, 2018: Інститут проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України, С. 107-110.

[3]. С. Гільгурт, "Побудова асоціативної пам'яті на цифрових компараторах реконфігуровними засобами для вирішення задач інформаційної безпеки", *Электронное моделирование,* Т. 41, № 3, С. 59-80, 2019.

[4]. С. Гільгурт, "Побудова фільтрів Блума реконфігуровними засобами для вирішення задач інформаційної безпеки", *Безпека інформації,* Т. 35, № 1, С. 53-58, 2019.

[5]. К. Самофалов, А. Ромлинкевич, В. Валуйский, Ю. Каневский, М. Пиневич, *Прикладная теория цифровых автоматов*. Київ: Вища шк. Головное изд-во, 1987.

[6]. R. Keller, *Computer Science: Abstraction to Implementation*. Harvey Mudd College, 2001, p. 630.

[7]. B. Smyth, *Computing Patterns in Strings*. Essex: Pearson Addison Wesley, 2003, 496 с.

[8]. A. Aho, M. Corasick, "Efficient String Matching: An Aid to Bibliographic Search", *Communications of the ACM,* vol. 18, no. 6, pp. 333-340, 1975.

[9]. W. Jiang, Y. Yang, V. Prasanna, "Scalable multi-pipeline architecture for high performance multi-pattern string matching", *24th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2010*, Atlanta, GA, 2010.

[10]. T. Song, W. Zhang, D. Wang, Y. Xue, "A memory efficient multiple pattern matching architecture for network security", *27th Ieee Conference on Computer Communications (Infocom), Vols 1-5,* Proceedings Paper pp. 673-681, 2008.

[11]. C. Lin, Y. Tai, S. Chang, "Optimization of pattern matching algorithm for memory based architecture C3", ANCS'07. Proceedings of the 2007 ACM Symposium on Architecture for Networking and Communications, *3rd ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2007*, Orlando, FL, pp. 11-16, 2007.

[12]. C. Lin, S. Chang, "Efficient Pattern Matching Algorithm for Memory Architecture", *Ieee Transactions on Very Large Scale Integration (Vlsi) Systems,* Article vol. 19, no. 1, pp. 33-41, Jan 2011.

[13]. Q. Wang, V. Prasanna, I. Soc, "Multi-Core Architecture on FPGA for Large Dictionary String Matching", *Proceedings of the 2009 17th Ieee Symposium on Field Programmable Custom Computing Machines,* Proceedings Paper, pp. 96-103, 2009.

[14]. J. Lunteren, "High-performance pattern-matching for intrusion detection", *25th Ieee International Conference on Computer Communications, Vol. 1-7, Proceedings Ieee Infocom 2006,* Proceedings Paper pp. 1409-1421, 2006.

[15]. H. Lu, K. Zheng, B. Liu, X. Zhang, Y. Liu, "A memory-efficient parallel string matching architecture for high-speed intrusion detection", *Ieee Journal on Selected Areas in Communications,* Article vol. 24, no. 10, pp. 1793-1804, Oct 2006.

[16]. L. Tan, T. Sherwood, I. Soc, "A high throughput string matching architecture for intrusion detection and prevention", *32nd International Symposium on Computer Architecture*, Madison, WI, Jun 04-08 2005, LOS ALAMITOS: Ieee Computer Soc, in Conference Proceedings Annual International Symposium on Computer Architecture, pp. 112-122, 2005.

[17]. P. Piyachon, Y. Luo, "Efficient memory utilization on network processors for deep packet inspection", *2nd ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2006*, San Jose, CA, pp. 71-80, 2006.

[18]. K. Huang, D. Zhang, "A Byte-Filtered String Matching Algorithm for Fast Deep Packet Inspection", Proceedings of the 9th International Conference for Young Computer Scientists, Vols 1-5, Proceedings Paper, pp. 2073-2078, 2008.

[19]. H. Jung, Z. Baker, V. Prasanna, "Performance of FPGA implementation of bit-split architecture for intrusion detection systems", *20th IEEE International Parallel and Distributed Processing Symposium, IPDPS 2006*, 2006, vol. 2006: IEEE Computer Society.

[20]. D. Pao, W. Lin, B. Liu, "Pipelined architecture for multi-string matching," *IEEE Computer Architecture Letters,* vol. 7, no. 2, pp. 33-36, 2008.

ПОСТРОЕНИЕ КОНЕЧНЫХ АВТОМАТОВ РЕКОНФИГУРИРУЕМЫМИ СРЕДСТВАМИ ДЛЯ РЕШЕНИЯ ЗАДАЧ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Противодействие угрозам информационной безопасности требует осуществления мероприятий различного плана – организационных, технических, криптографических и др. Сетевые системы обнаружения вторжений, антивирусы и прочие подобные средства технической защиты, работа которых основана на использовании сигнатур, должны решать в реальном времени вычислительно сложную задачу множественного распознавания строк, которая в отличие от одиночного распознавания подразумевает одновременный поиск во входных данных большого количества образцов. Программные решения уже не справляются с этой проблемой в связи с устойчивым ростом объема сетевого трафика, количества и сложности атак. Поэтому все большее распространение получают аппаратные решения с использованием реконфигурируемых устройств на базе ПЛИС типа FPGA, которые сочетают в себе близкую к аппаратной производитель-

ность с гибкостью программного обеспечения. На сегодня наиболее распространены три подхода к построению аппаратных схем множественного распознавания, работа которых основана на использовании: ассоциативной памяти, фильтра Блума и конечных автоматов. Каждый из подходов имеет свои преимущества и недостатки. Эффективное построение все более сложных сигнатурных средств технической защиты информации, которые бы отвечали требованиям оптимального функционирования в зависимости от внешних условий, невозможно без всестороннего анализа свойств и специфических черт каждого из подходов. В этом исследовании с целью повышения эффективности создаваемых на базе ПЛИС средств информационной безопасности проанализированы преимущества и недостатки третьего из перечисленных подходов. На основе анализа мирового опыта использования конечных автоматов также исследованы особенности их реализации на ПЛИС, возникающие проблемы и пути их решения.

Ключевые слова: защита информации, сигнатурный анализ, ПЛИС, конечный автомат, алгоритм Ахо-Корасик, эффективность.

CONSTRUCTING DETERMINISTIC FINITE AUTOMATA BY RECONFIGURABLE MEANS FOR SOLVING INFORMATION SECURITY TASKS

Computer security requires the implementation of different measures - organizational, technical, cryptographic, etc. Network intrusion detection systems, antiviruses and other similar tools of technical protection based on signatures should solve in real time the computationally complex multi-pattern string matching task, which is a specific type of string matching functionality performed in DPI systems to search an input stream for a set of patterns rather than a single pattern. Due to rising traffic rates, increasing number and sophistication of attacks and the collapse of Moore's law for sequential processing, traditional software solutions can no longer meet the high requirements of today's security challenges. Therefore, hardware approaches are proposed to accelerate pattern matching. Combining the flexibility of software and the near-ASIC performance, reconfigurable hardware devices based on Field Programmable Gate Arrays (FPGA) have become increasingly popular for this purpose. There are three main approaches to fulfill the computation-intensive multi-pattern string matching task using FPGA. The techniques of these approaches are: content addressable memory, Bloom filter and Aho-Corasick Algorithm. Each approach has its advantages and disadvantages. Effective construction of more and more complex signature-based tools of technical secure protection is impossible without a comprehensive analysis of the properties and specific features of every approach. In this study, in order to improve the efficiency of information security tools created on the basis of the FPGA, the main features of the third of the listed approaches are analyzed. Based on the analysis of the state-of-the-art finite automata realizing Aho-Corasick algorithm, the specialty of their implementation on FPGAs, encountering problems and ways to solve them are also investigated.

Keywords: information security, multi-pattern string matching, FPGA, finite automaton, Aho-Corasick algorithm, efficiency.

Гільгурт Сергій Якович, кандидат технічних наук, старший науковий співробітник, старший науковий співробітник відділу Теорії моделювання Інституту проблем моделювання в енергетиці ім. Г.Є. Пухова НАН України.
E-mail: hilgurt@ipme.kiev.ua.
Orcid ID: 0000-0003-1647-1790.
Гильгурт Сергей Яковлевич, кандидат технических наук, старший научный сотрудник, старший научный сотрудник отдела Теории моделирования Института проблем моделирования в энергетике им. Г.Е. Пухова НАН Украины.
Hilgurt Serhiy, Candidate of Technical Sciences, Senior Researcher, Senior Researcher of Department of Modelling Theory, Pukhov Institute for Modelling in Energy Engineering of NAS of Ukraine.